

Zadanie 5

Treść

Ułóż algorytm, który dla danego grafu $G = (V, E)$ oraz liczby naturalnej k znajdzie możliwie największy podzbiór $V' \subseteq V$, taki że dla każdego wierzchołka $v \in V'$ zachodzi:

$$|\{u \in V' : \{v, u\} \in E\}| \geq k \text{ oraz}$$

$$|\{u \in V' : \{v, u\} \notin E\}| \geq k$$

Wyjaśnienie

Celem jest znalezienie największego (w sensie liczby wierzchołków) podzbioru V' grafu G , takiego że każdy wierzchołek v w tym podzbiorze V' ma:

1. Co najmniej k **sąsiadów** należących do V' .
2. Co najmniej k **nie-sąsiadów** należących do V' (czyli wierzchołków z $V' \setminus v$ z którymi v nie tworzy krawędzi w G).

Algorytm

Niech V_{cur} oznacza zbiór aktualnie rozważanych (jeszcze nieusuniętych wierzchołków). Początkowo $V_{\text{cur}} = V$. Dla każdego wierzchołka $v \in V_{\text{cur}}$ będziemy śledzić jego stopień $d_{\text{cur}}(v)$ (liczbę sąsiadów w V_{cur}).

1. Utwórz kubelki $\text{KUB}[i]$ dla i od 0 do $|V| - 1$. Dla każdego wierzchołka $v \in V$ oblicz jego początkowy stopień $d(v)$ w grafie G i umieść v w kubelku $\text{KUB}[d(v)]$.
2. Utwórz pustą kolejkę Q . Dla każdego wierzchołka $v \in V_{\text{cur}}$:
 - i. Jeśli $d_{\text{cur}}(v) < k$ (za mało sąsiadów w V_{cur}), dodaj v do Q .
 - ii. Jeśli $|V_{\text{cur}}| - 1 - d_{\text{cur}}(v) < k$ (za mało nie-sąsiadów w V_{cur}), dodaj v do Q (jeśli nie został już dodany).

Oznacz wierzchołki dodane do Q jako “do usunięcia”, aby uniknąć duplikatów.

3. Dopóki kolejka Q nie jest pusta:
 - i. Wyjmij wierzchołek v z Q . Oznacz v jako “usunięty” i usuń go z V_{cur} (zmniejszając tym samym $|V_{\text{cur}}|$).
 - ii. Dla każdego sąsiada u wierzchołka v (takiego, który jest w V_{cur} i nie jest “usunięty” ani “do usunięcia”):
 - a. Przenieś u z $\text{KUB}[d_{\text{cur}}(u)]$ do $\text{KUB}[d_{\text{cur}}(u) - 1]$. Zaktualizuj $d_{\text{cur}}(u) \leftarrow d_{\text{cur}}(u) - 1$.
 - b. Jeśli $d_{\text{cur}}(u) < k$ (teraz u ma za mało sąsiadów), dodaj u do Q i oznacz jako “do usunięcia”.
 - iii. Po usunięciu v i zaktualizowaniu wszystkich sąsiadów, rozmiar V_{cur} zmalał. Może to spowodować, że niektóre wierzchołki mają teraz za mało nie-sąsiadów.
Dla każdego wierzchołka w znajdującego się w kubelku $\text{KUB}[|V_{\text{cur}}| - k]$:
 - jeśli w jest w V_{cur} i nie jest “do usunięcia”, dodaj w do Q i oznacz jako “do usunięcia”. (Warunek $|V_{\text{cur}}| - 1 - d_{\text{cur}}(w) < k$ będzie automatycznie spełniony dla $d_{\text{cur}}(w) = |V_{\text{cur}}| - k$).
4. Zbiór V' to wszystkie wierzchołki, które nie zostały oznaczone jako “usunięte” (czyli pozostałe V_{cur}).

Analiza złożoności

1. Inicjalizacja kubeków

$O(V + E)$

Obliczenie stopni wszystkich wierzchołków i umieszczenie ich w kubkach.

2. Inicjalizacja kolejki

$O(V)$

Przejrzenie wszystkich $|V|$ wierzchołków i ewentualne dodanie ich do kolejki.

3. Pętla główna

Pętla wykonuje się co najwyżej $|V|$ razy, ponieważ każdy wierzchołek może być dodany do kolejki i usunięty co najwyżej raz.

i. Wyjęcie z kolejki, oznaczenie

$O(V)$

Obie operacje wykonują się w czasie $O(1)$.

ii. Aktualizacja sąsiadów

Gdy wierzchołek v jest usuwany, przeglądamy jego sąsiadów. Każda krawędź $\{v, u\}$ jest brana pod uwagę co najwyżej dwa razy w całym algorytmie (raz gdy v jest usuwane i raz gdy u jest usuwany).

Każdy wierzchołek jest dodawany do kolejki co najwyżej raz.

Summaryczne koszty

- Aktualizacja stopni i przenoszenie między kubkami
- Dodawanie sąsiadów do kolejki

$O(E)$

$O(V)$

iii. Sprawdzenie warunku nie-sąsiadów

$O(V)$

Każdy wierzchołek może zostać dodany do kolejki co najwyżej raz oraz sprawdzamy tylko jeden kubek.

Całkowita złożoność czasowa algorytmu wynosi $O(V + E)$.

Złożoność pamięciowa to $O(V + E)$ na przechowanie grafu, kubków i kolejki.

Dowód poprawności

Dowód opiera się na niezmienniku pętli. Niech V_{cur} oznacza zbiór wierzchołków nieusuniętych.

Na początku każdej iteracji głównej pętli, każdy wierzchołek $w \in V_{\text{cur}}$, który nie znajduje się w kolejce Q , spełnia oba warunki zadania względem aktualnego zbioru V_{cur} :

1. Liczba sąsiadów $w \in V_{\text{cur}}$ (oznaczane jako $d_{\text{cur}}(w)$) jest $\geq k$.
2. Liczba nie-sąsiadów w w V_{cur} (to jest $|V_{\text{cur}}| - 1 - d_{\text{cur}}(w)$) jest $\geq k$.

Inicjalizacja

Przed pierwszą iteracją pętli (po inicjalizacji kolejki:

- Wszystkie wierzchołki $v \in V$ (bo $V_{\text{cur}} = V$ na tym etapie) zostały sprawdzone.
- Te, które nie spełniały warunku liczby sąsiadów lub nie-sąsiadów, zostały dodane do kolejki Q .

Zatem każdy wierzchołek $w \in V_{\text{cur}}$ niebędący w Q musi spełniać oba warunki.

Utrzymanie

Założmy, że niezmiennik jest prawdziwy na początku iteracji. Wierzchołek v jest wyjmowany z Q i usuwany z V_{cur} .

1. Usunięcie

v nie jest już w V_{cur} , więc nie musi spełniać niezmiennika. Rozmiar V_{cur} maleje o 1.

2. Aktualizacja sąsiadów

Dla każdego sąsiada u wierzchołka v (który jest w V_{cur}):

- i. Jego stopień $d_{\text{cur}}(u)$ maleje o 1.
- ii. Jeśli nowy $d_{\text{cur}}(u)$ spadnie poniżej k , u jest dodawany do Q .

3. Sprawdzenie nie-sąsiadów

Liczba nie-sąsiadów wynosi teraz $|V_{\text{cur}}|_{\text{nowy}} - 1 - d_{\text{cur}}(w)$. Jeśli ta wartość jest mniejsza od k , w jest dodawany do Q . Algorytm identyfikuje takie wierzchołki dla których liczba nie-sąsiadów to $k - 1$ (poprzez odwołanie do odpowiedniego kubelka).

Po tych operacjach każdy wierzchołek $w \in V_{\text{cur}}$, który nie jest w Q , musi mieć $d_{\text{cur}}(w) \geq k$ (bo inaczej zostałby dodany w poprzedniej iteracji lub w kroku ii.) oraz $|V_{\text{cur}}| - 1 - d_{\text{cur}} \geq k$ (bo inaczej zostałby dodany w poprzedniej iteracji lub w kroku iii.).

Terminacja

Pętla kończy się, gdy kolejka Q jest pusta. Zgodnie z niezmiennikiem, w tym momencie każdy wierzchołek w w ostatecznym zbiorze V_{cur} (czyli V') nie jest w Q i nie jest usunięty. Dlatego każdy $w \in V'$ spełnia oba warunki:

1. $d_{V'}(w) \geq k$
2. $|V'| - 1 - d_{V'}(w) \geq k$

Jest to dokładnie definicja wymaganego podzbioru.

■

Maksymalność

Algorytm usuwa wierzchołki iteracyjnie. Wierzchołek jest usuwany tylko wtedy, gdy narusza co najmniej jeden z warunków w kontekście aktualnie nieusuniętych wierzchołków.

Jeśli wierzchołek v został usunięty, oznacza to, że nie mógłby być częścią żadnego poprawnego rozwiązania V_{opt} będącego podzbiorem aktualnego V_{cur} (w momencie przed usunięciem v), ponieważ v naruszałby warunki w V_{opt} (gdyż $V_{\text{opt}} \subseteq V_{\text{cur}}$ i warunki są monotoniczne w sensie, że usunięcie innych wierzchołków nie poprawi sytuacji v). Każdy wierzchołek, który mógłby należeć do jakiegokolwiek rozwiązania, nie zostanie usunięty. Zatem znaleziony zbiór V' jest największym możliwym podzbiorem.